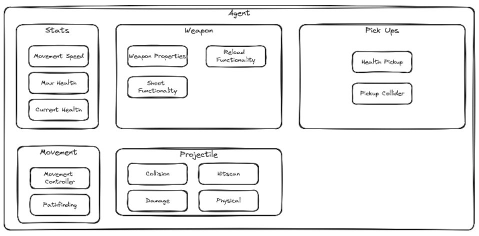
Project 2-1 ML Group 18

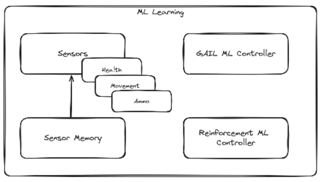
Introduction

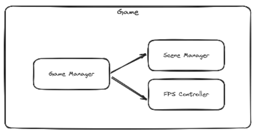
This project aims to develop a simplified 3D First-Person Shooter (FPS) game in which two AI-controlled agents compete against each other or compete against a human player.

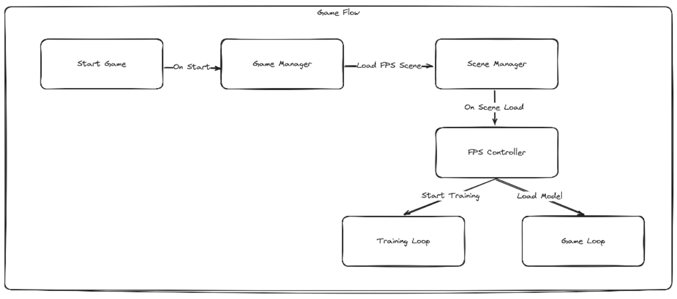
The game will incorporate fundamental mechanics commonly found in FPS games, including guns, ammo, and health systems, providing a foundational exploration of autonomous agent behaviour in an interactive gaming environment.

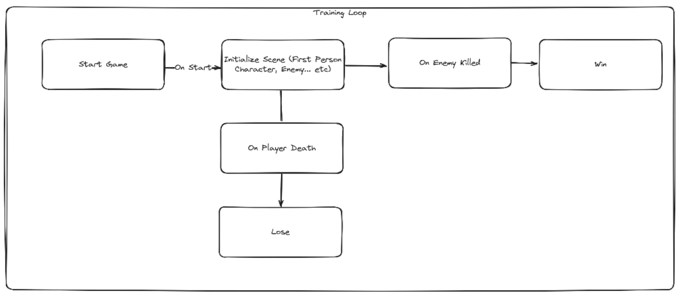
Block Diagram

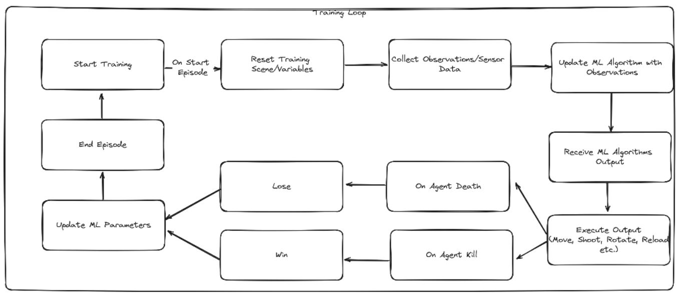











Training Loop
Start Training
On Start Episode
Reset Training Scene/Variables
Collect Observations/Sensor Data
Update ML Algorithm with Observations
Receive ML Algorithms Output
Execute Output (Move, Shoot, Rotate, Reload etc.)
On Agent Death
On Agent Kill
Lose
Win
Update ML Parameters
End Episode

Software Component Preparation

Unity:

The game scene, visuals, colliders, gameobjects and components will be set up in the Unity editor.

C#:

Agents, Sensors, Pickups, Managers and Controllers will be implemented in C#.
As well as the various logic and behaviour for Agents.
Implementing the custom logic for ML algorithms will be implemented in C#. Such as what awards points in a Reinforcement Learning algorithm, when a training episode begins and ends… etc.

Python:

No custom Python code is planned as the ML algorithms researched are natively implemented in the ML package.

Task Description

- Create Basic FPS Agent
    - Create a basic agent controller that will manage the logic for movement, shooting and stats.
- Create Agent Movement
    - Create movement controllers that will move and rotate an agent given input.
- Create Weapon System
    - Create a weapon with a visual model and behaviour for shooting projectiles and reloading ammo.
- Create Projectile System
    - Create physical and hit-scan projectiles with collision and damage.
- Create Health Pickups
    - Create an entity that agents can collide and pick up. Create a health pickup that will restore an agent’s health when picked up.
- Create Agent Stats
    - Create health and movement speed stats for agents
- Create Agent Vision Sensor
    - Create a sensor that detects enemies within an agent’s field of view. Records the entity detected and its position.
- Create Agent Sound Sensor
    - Create a sensor that detects noise produced from various actions such as moving, shooting

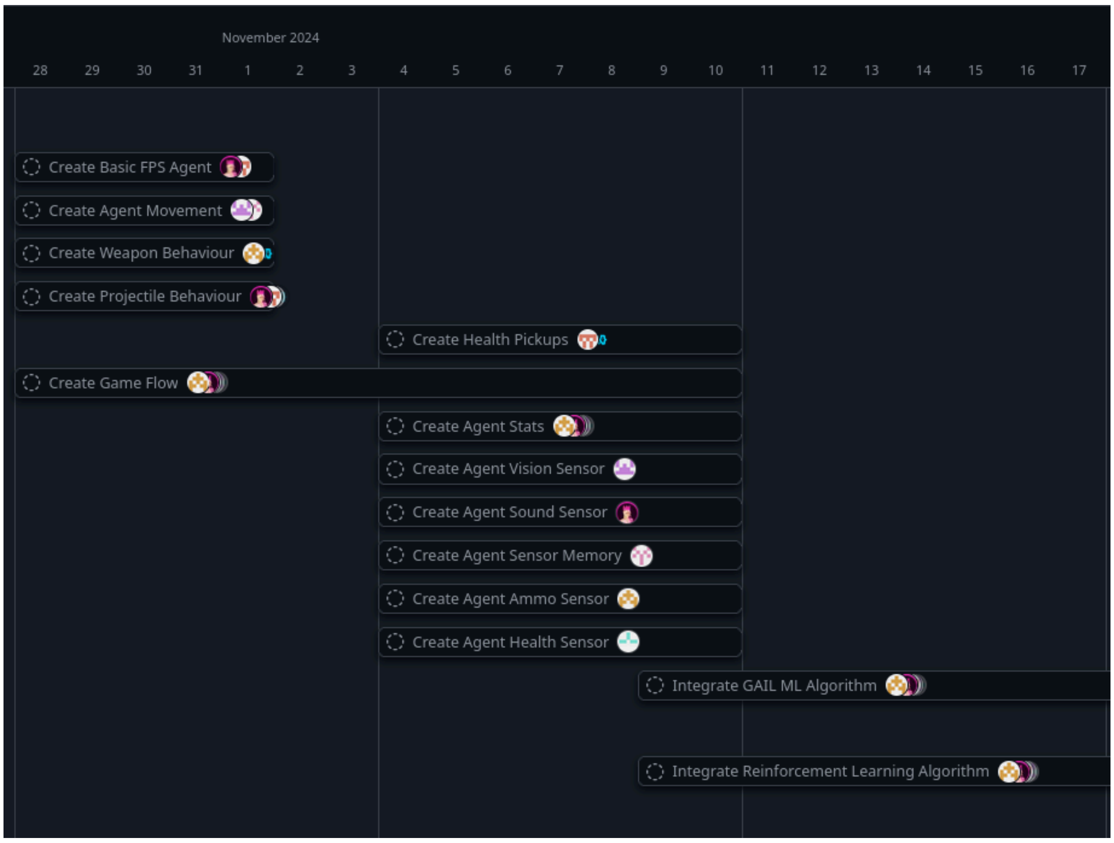
    and reloading. Records the estimated position and action of the sound.

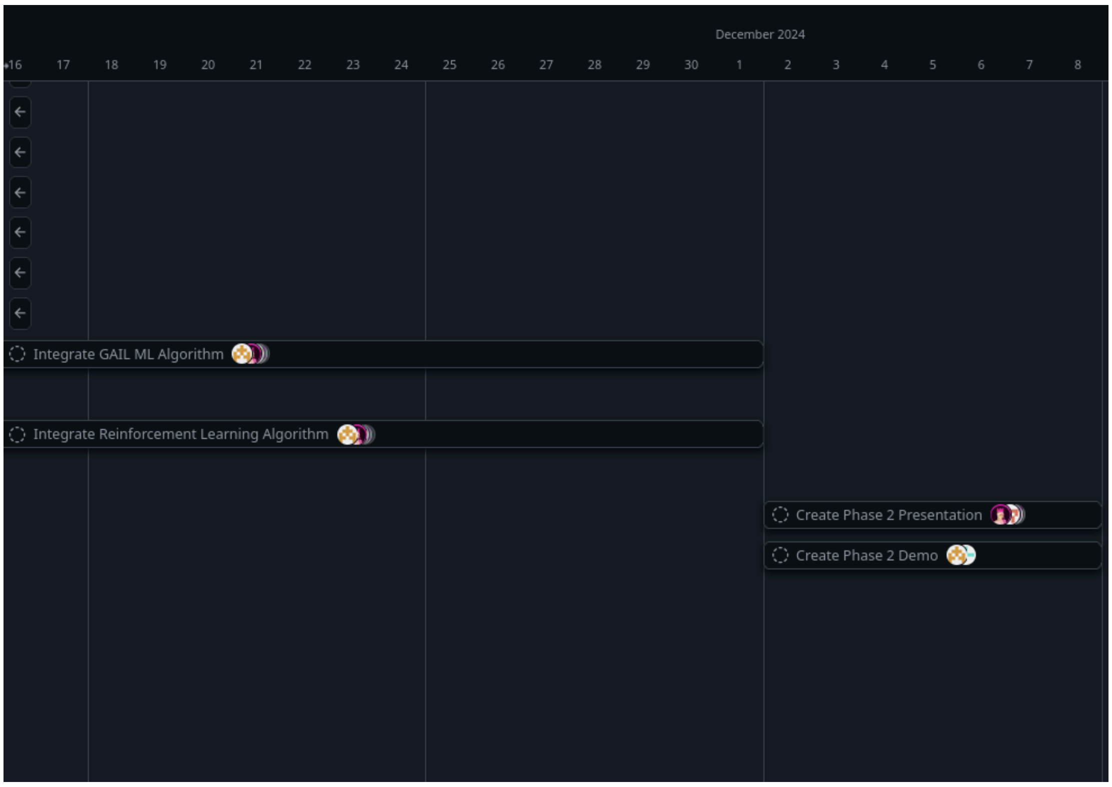
- Create Agent Ammo Sensor
    - Create a sensor that detects the current ammo of an agent. Records the current and maximum ammo.
- Create Agent Health Sensor
    - Create a sensor that detects the current health of an agent. Records the current and maximum health.
- Create Agent Sensor Memory
    - Add memory to other sensors to maintain a history of recently sensed data.
- Create Game Flow
    - Create a class to manage the initialization of the game for the type of AI model to use, saving and loading AI Model and AI vs AI or AI vs Player behaviour.
- Integrate Reinforcement Learning Algorithm
    - Integrate a Reinforcement Learning algorithm with Agent behaviour to record positive or negative actions an agent takes.
- Integrate GAIL ML Algorithm
    - Integrate a Generative Adversarial Imitation Learning algorithm. Define initial behaviour to imitate. Implement logic for cost function.
- Experiment with Machine Learning Algorithms
    - Tweak parameters in ML algorithms and analyse results on agents’ behaviours.

- Profile GAIL
    - Analyse the performance and efficiency of the GAIL algorithm
- Profile Reinforcement Learning
    - Analyse the performance and efficiency of the Reinforcement Learning algorithm
- Define Report Content
    - Determine what we will include in the report
- Write Report
    - Write a report that outlines our whole project process from beginning to end and presents our findings.

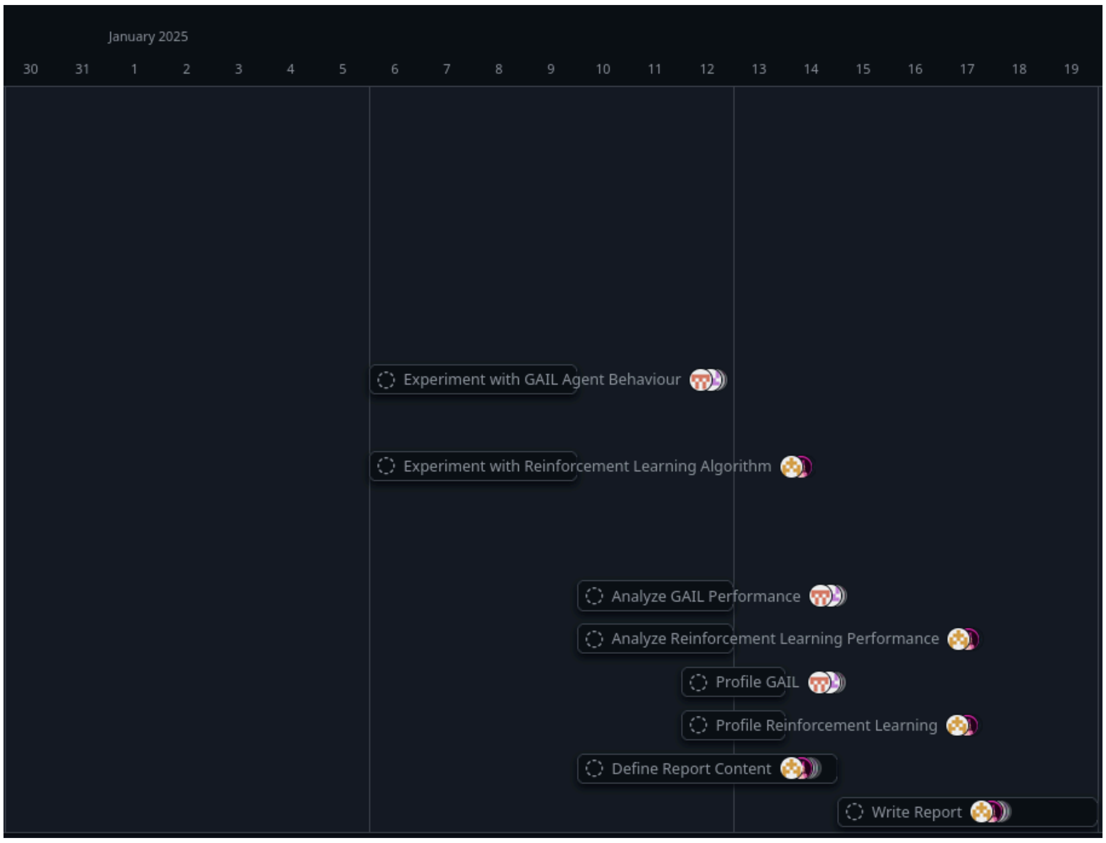
Gantt Chart

Phase 2





Phase 3


January 2025
30
31
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Experiment with GAIL Agent Behaviour
Experiment with Reinforcement Learning Algorithm
Analyze GAIL Performance
Analyze Reinforcement Learning Performance
Profile GAIL
Profile Reinforcement Learning
Define Report Content
Write Report

Risk Analysis

Custom Scene:

Problem:

Building our custom first person shooter scene comes with its own risks. One of them being that the development of the scene might take too long and further development be halted as a result of this.

Solution:

To solve this issue we will revert to the example scenes provided and continue development from there.

Custom Behaviour:

Problem:

Implementing the game functionalities such as agent movement and weapon behaviours might prove too challenging, in that case we will simplify the game to an extent where it is easy enough to create the scene.

Solution:

To solve this issue we will simplify the behaviour, such as removing weapons and health.

ML Implementation:

Problem:

Implementing custom behaviour for agents using RL and GAIL algorithms might prove too challenging. That integrating ML with the logic for movement and shooting is too difficult to implement within the given time frame or is too advanced.

Solution:

To solve this issue we will look for alternate ML algorithms that will simplify the integration process. If no alternative algorithms are found then this is not an issue with the ML Implementation but rather with the Custom Behaviour.

Optimized AI:

Problem:

Implementing ML to control Agents might prove too optimal. Such that an AI will always win against a Player.

Solution:

To solve this issue we will experiment and tweak the ML algorithm’s parameters.